ICASE

A MULTI-COLOR SOR METHOD FOR PARALLEL COMPUTATION

Loyce Adams

and

James M. Ortega

Report No. 82-9

April 8, 1982

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the

UNIVERSITIES SPACE USRA RESEARCH ASSOCIATION

A MULTI-COLOR SOR METHOD FOR PARALLEL COMPUTATION

Loyce Adams
University of Virginia

James M. Ortega
University of Virginia

ABSTRACT

This paper considers a generalization of the classical red/black ordering of grid points for finite difference or finite element discretizations shown to be effective in the implementation of the SOR iteration method on vector or parallel computers. Examples are given of various orderings for different discretizations and implementation on the CDC Cyber 203/205 and the Finite Element Machine is discussed.

This research was sponsored by NASA Contract No. NAS1-46. The work of the second author was partially supported under NASA Contract No. NAS1-16394 while he was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665.

A Multi-Color SOR Method for Parallel Computation

L. Adams and J. Ortega

1. Introduction

We are concerned in this paper with the solution of a sparse nxn linear system of equations

$$A\underline{x} = \underline{b} \tag{1.1}$$

by iterative methods, especially SOR type methods, on parallel arrays or vector computers. As opposed to the Jacobi iteration, which has rather ideal properties for parallel computation, the SOR method is essentially a sequential method. However, several authors (e.g. Hayes [1974], Lambiotte [1975]) have observed that if (1.1) arises from a five-point finite difference discretization of Poisson's equation and the equations are ordered according to the classical Red/Black partitioning of the grid points then an SOR sweep may be carried out, in essence, by two Jacobi sweeps, one on the equations corresponding to the red points and one for the equations corresponding to the black points. Thus, in this case, the SOR method can be effectively implemented on vector or parallel computers.

This strategy does not work, however, for higher order finite difference or finite element discretizations or for more general elliptic equations which contain mixed partial derivative terms. In these cases, it is necessary to generalize the Red/Black partitioning of the grid points to a "multi-color" partitioning; for example, a three color partitioning, say Red/Black/Green, might give the desired result. In general, the number of colors necessary will depend on the connectivity pattern of the grid points. If p colors are used, an SOR sweep can be implemented by p Jacobi sweeps, one for each set of equations associated with a given color. For vector computers, this reduces the effective vector length to O(n/p) while for parallel arrays it is necessary that each processor hold a multiple of p equations. This multiple will be determined by the particular discretization. Clearly, there will be a point of diminishing returns as p increases but for most differential equations and discretizations of interest it seems that no more than 6 colors will suffice and for the size of n we have in mind ($n \approx 10{,}000+$), the multi-color strategy can be very effective.

We note that multi-color orderings for SOR have been used before (see Young [1971]) but, to the best of our knowledge, have not been used in the context of parallel computation.

In the next section, we describe the method in more detail and in Section 3 we discuss some of the implementation questions for both vector computers and parallel arrays. We do not address the many other problems in the successful use of the SOR iteration, especially the problem of determining an optimum relaxation parameter.

2. Multi-Color Orderings

For concreteness, we consider first an elliptic equation of the form

$$u_{xx} + au_{xy} + u_{yy} = f \tag{2.1}$$

on the unit square with Dirichlet boundary conditions where a is a given constant and f is a given function of x and y. We discretize (2.1) with the usual second-order finite difference approximations (see, e.g., Forsythe and Wasow [1960]) which give the difference equations

$$\begin{aligned} &u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij} + \frac{a}{4}[u_{i+1,j+1} \\ &- u_{i-1,j+1} + u_{i-1,j-1} - u_{i+1,j-1}] = h^2 f_{ij} \end{aligned} \tag{2.2}$$

where h is the spacing between grid points, $i,j = 1..N$ where $h(N+1)=1$, u_{ij} denotes the approximate solution at the i,jth grid point, and $f_{ij} = f(ih,jh)$. Now partition the grid points by the Red/Black scheme, as indicated by Figure 1, and then number the grid points in each class from left to right, bottom to top.

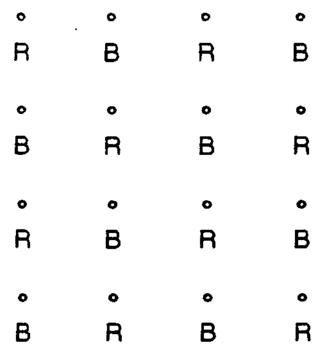


Figure 1. Red/Black Ordering

If a=0, so that (2.1) is just Poisson's equation, then it is well-known (see e.g. Young [1971]) that the difference equations (2.2) may be written in the partitioned matrix form

$$\begin{bmatrix} D & B \\ B^T & D \end{bmatrix} \begin{bmatrix} \underline{u}_r \\ \underline{u}_b \end{bmatrix} = \begin{bmatrix} \underline{f}_r \\ \underline{f}_b \end{bmatrix} \tag{2.3}$$

where D is a diagonal matrix and $\underline{u}_r$ and $\underline{u}_b$ denote the vectors of unknowns associated with the red and black grid points respectively. The Gauss-Seidel iteration for (2.3) may be written as

$$\begin{aligned} D\underline{u}_r^{k+1} &= -B\underline{u}_b^k + \underline{f}_r \\ D\underline{u}_b^{k+1} &= -B^T\underline{u}_r^{k+1} + \underline{f}_b \end{aligned} \tag{2.4}$$

and each part of (2.4) can then be effectively implemented in a parallel fashion, with the introduction of the SOR parameter causing no problem.

If a $\neq$ 0, the form (2.3) of the difference equations is still valid although D is no longer a diagonal matrix and the Gauss-Seidel step (2.4) is no longer easily implementable in a parallel fashion. The problem is that unknowns corresponding to red points are coupled to each other in (2.3) (and black points to each other also) whereas when a=0 , they completely uncouple. Thus we wish to introduce another partitioning of the grid points for which unknowns within each subset of the partitioning are uncoupled. If we consider the grid point stencil for (2.2), shown in Figure 2,

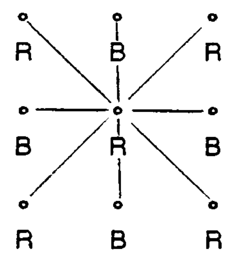


Figure 2. Stencil for (2.2)

with the Red/Black ordering, we see that the center Red point is connected to the Red points at the four corners. If, however, we use four subsets of grid points, labeled red, black, white, orange, we can ensure that each center point connects with only points of different colors. A suitable coloring pattern for this is illustrated in Figure 3.

R	B	W	O	R	B	W	O
W	O	R	B	W	O	R	B
R	B	W	O	R	B	W	O
W	O	R	B	W	O	R	B

Figure 3. Four color partitioning of the gridpoints

In this case, the system (2.2) can be written in a partitioned form analogous to (2.3) as

$$\begin{bmatrix} D_1 & B_{12} & B_{13} & B_{14} \\ B_{21} & D_2 & B_{23} & B_{24} \\ B_{31} & B_{32} & D_3 & B_{34} \\ B_{41} & B_{42} & B_{43} & D_4 \end{bmatrix} \begin{bmatrix} \underline{u}_r \\ \underline{u}_b \\ \underline{u}_w \\ \underline{u}_o \end{bmatrix} = \begin{bmatrix} \underline{f}_r \\ \underline{f}_b \\ \underline{f}_w \\ \underline{f}_o \end{bmatrix} \tag{2.5}$$

where D_1, D_2, D_3, and D_4 are diagonal matrices. The Gauss-Seidel iteration in terms of (2.5) is then

$$\begin{aligned} D_1\underline{u}_r^{k+1} &= -B_{12}\underline{u}_b^k - B_{13}\underline{u}_w^k - B_{14}\underline{u}_o^k + \underline{f}_r \\ D_2\underline{u}_b^{k+1} &= -B_{21}\underline{u}_r^{k+1} - B_{23}\underline{u}_w^k - B_{24}\underline{u}_o^k + \underline{f}_b \end{aligned} \tag{2.6}$$

with similar equations for $\underline{u}_w^{k+1}$ and $\underline{u}_o^{k+1}$. Since the D_i are diagonal, (2.6) is easily implementable on vector or parallel architectures.

A variety of other connectivity patterns arise from either finite difference or finite element discretizations. Two of the more common are illustrated by their stencils in Figure 4,

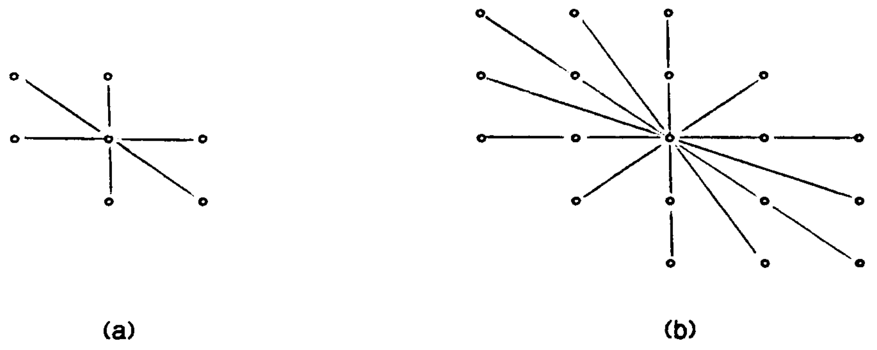


Figure 4. Common finite element stencils

in which (a) arises, for example, from finite element discretization by piecewise linear functions over triangular subregions and (b) by piecewise quadratic functions. In case (a), three colors are necessary and sufficient to achieve the desired decoupling while in case (b) six colors are required. The coloring patterns for the two cases are illustrated in

Figure 5.

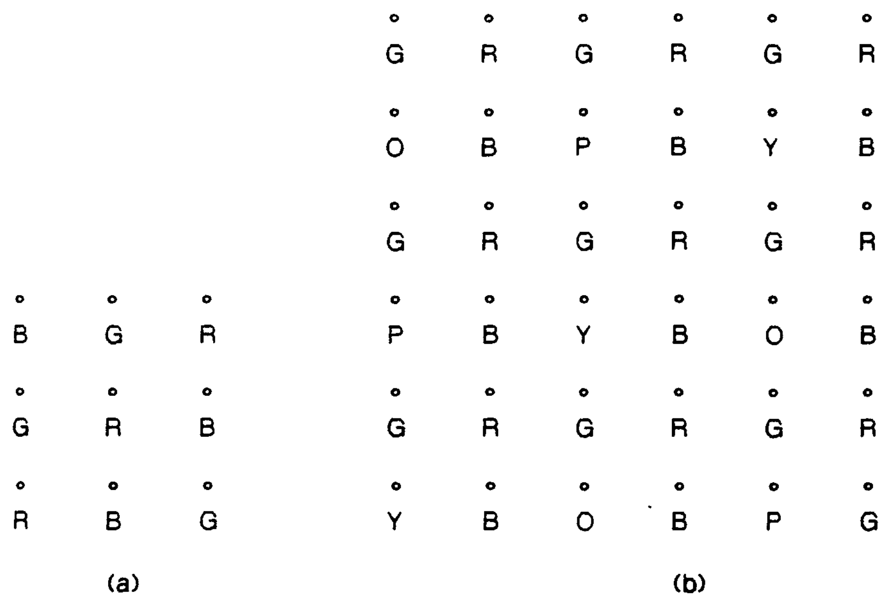


Figure 5. Three and six color partitions

In both cases, the color patterns repeat beyond the subregions illustrated.

A variety of other examples could be given. Provided that the domain of the differential equation is a rectangle or other regular two or three dimensional region and the discretization stencil is repeated at each grid, it is usually evident how to color the grid points to achieve the desired result. However, for arbitrary discretizations and/or irregular regions there is at present no algorithm to carry out the coloring.

3. Implementation Considerations

We discuss briefly in this section some of the implementation considerations of the multi-color SOR method on vector computers and parallel arrays. For concreteness, we will use the CDC CYBER 203/205 as an example of the former and the Finite Element Machine at NASA's Langley Research Center as an example of the latter.

On the CYBER 203/205, vectors consist of contiguous storage locations and the efficiency of the vector operations is strongly dependent on vector length. Maximum efficiency is achieved for very long vectors.

For vectors of length 1000 around 90% efficiency is obtained, but this drops to approximately 50% or less for vectors of length 100 and less than 10% for length 10. Hence, we would like to keep vector lengths on the order of 1000 or more whenever possible.

Consider, for example, the difference equations (2.2) and suppose that h=.01 so that N=99 and $n=N^2 \approx 10^4$. The implementation of Jacobi's method on this problem can be done in a straightforward way using vectors of length N, corresponding to the unknowns in each row of grid points. It is desirable, however, to work with vectors of length order n and it is possible to achieve this by considering the boundary values to be unknowns and ordering all the grid points, including the boundary points, from left to right, bottom to top and then applying the Jacobi iteration to the corresponding vector of length $(N+2)^2$ of unknowns. The boundary values, of course, cannot be changed by the iteration and this is prevented by use of the control vector feature on the 203/205 which allows suppression of storage of updated values into the boundary locations. (See, e.g. Lambiotte [1975] or Ortega and Voigt [1977] for more details on this procedure.) Since the calculation of new values corresponding to the boundary points is superfluous, this introduces an inefficiency of approximately 4% for N=99 but allows almost full efficiency of the vector operations.

For the Gauss-Seidel or SOR method for (2.2) we use the four-color ordering of Figure 3, and order the unknowns into four vectors corresponding to the grid points associated with the four colors. The matrix-theoretic description (2.6) of the Gauss-Seidel iteration is then implemented by four separate Jacobi sweeps, one for each color. As above, the boundary values are considered as unknowns and then updated values suppressed on storage. Since the vector lengths are now on the order of 2500, the corresponding vector operations will run at about 95% efficiency. The introduction of the SOR parameter causes no difficulty.

We turn now to parallel arrays. The Finite Element Machine is a prototype array of 36 microprocessors, arranged in a 6x6 grid. Each processor is connected to eight nearest neighbors, as illustrated in

Figure 6.

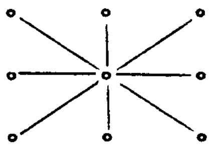


Figure 6. Processor Interconnections on the Finite Element Machine

and there is also a global bus that connects all processors. Further details, which do not concern us here, may be found in Jordan [1978] and the references therein.

Our primary goal in the implementation of the multi-color SOR method on the Finite Element Machine, or on a similar array with perhaps many more processors but limited processor to processor interconnections, is to keep as many processors as possible running at a given time. This, in turn, requires maximum use of the processor interconnections and minimum use of the global bus since contention for the bus will tend to introduce delays which cause processors to be idle.

Perhaps the primary consideration in the implementation is to ensure that each processor holds at least as many unknowns as a certain multiple of the number of colors where this multiple is the number of rows above the center point in the gridpoint interconnection stencil. Thus, for example, if we consider the gridpoint interconnection stencil of Figure 4(a) and the corresponding three color ordering of Figure 5(a), we would assign a minimum of 3 unknowns to each processor as illustrated in Figure 7(a). Similarly, for the stencil of Figure 4(b) and the corresponding six color ordering of Figure 5(b), we would assign a minimum of 12 unknowns to each processor as illustrated in Figure 7(b).

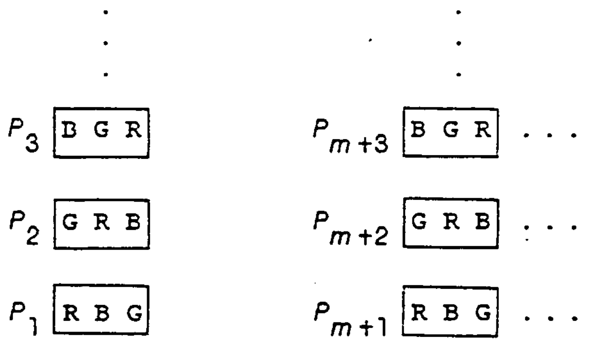
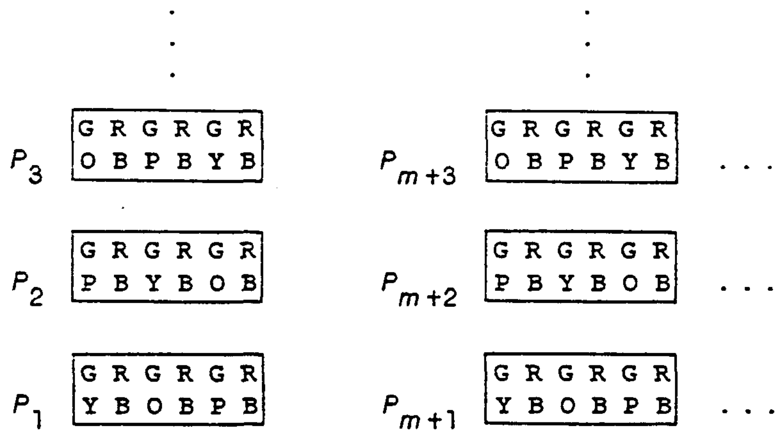


Figure 7(a). Processor Assignment



P_3 G R G R G R / O B P B Y B
P_2 G R G R G R / P B Y B O B
P_1 G R G R G R / Y B O B P B
P_{m+3} G R G R G R / O B P B Y B ...
P_{m+2} G R G R G R / P B Y B O B ...
P_{m+1} G R G R G R / Y B O B P B ...

Figure 7(b). Processor Assignment

In the simplest case of 36 processors and 108 grid points, with 108 corresponding unknowns, the assignment scheme of Figure 7(a) would be sufficient and the SOR method would be implemented by Jacobi operations, first on all the Red points, then the Black, then the Green. To carry out these Jacobi operations, current values of neighboring unknowns would be obtained either from the processor itself or a neighbor processor and no use of the global bus is necessary. Known boundary values would be stored in the processors which needed them. In any problem of interest, however, there would almost certainly be many more grid points and unknowns than processors. For the situation discussed above with three colors, we would assign unknowns in multiples of three to the processors. Similarly, for the grid point stencil of Figure 4(b) and corresponding six color pattern of Figure 5(b), we would assign unknowns in multiples of 12 to each processor.

The above assignment strategy would allow each processor to run without waiting except for two problems, synchronization and convergence. Consider a Jacobi operation on all the unknowns of a given color. The processors may complete their work on this operation in different times due to a number of factors: slightly different clock times in the processors; different memory access times, especially for those processors containing unknowns connected to boundary values; different numbers of unknowns assigned to processors and so on. To compensate for these possible differences in processing times, the computation can be synchronized by having each processor set a flag when it is done with its calculation on the current Jacobi operation and then wait for all other processors to complete. This synchronization, of course, introduces delays. Alternately, the processors can run asynchronously. In this case, the numerical iterations will tend to deviate from the true mathematical iteration, although the consequences of this may even be beneficial. (See, e.g. Baudet [1978] and the references therein for further discussion of asynchronous iterative methods.)

It is, of course, necessary to check for convergence of the iterative process. At the end of each SOR iteration, each processor can monitor the convergence of the unknowns assigned to it, probably by comparing the current and previous iterates. When the convergence criterion has been satisfied for all unknowns assigned to a given processor, that processor must continue the iteration until the convergence criterion is satisfied in all processors. Hence, the whole process will not terminate until all unknowns have satisfied the convergence criterion and towards the end of the process a portion of the processors may be doing unnecessary work. This seems to be an unavoidable inefficiency.

4. Summary and Conclusions

The multi-color SOR method described herein seems promising for vector and array processors although practical experience to date has been limited to a few numerical experiments on a four-processor version of the Finite Element Machine. It faces the usual difficulty with the SOR method of obtaining suitably good values of the overrelaxation parameter and for most applications of current interest for which a vector computer or large array would be used, there is little theory to help in this choice. For irregular regions, there is also the problem of processor assignment and coloring of the grid points; the processor assignment problem has been addressed by various authors (see, e.g. Bokhari [1979] and Gannon [1980]) but not in conjunction with the coloring problem.

REFERENCES

Baudet, G. [1978]. "Asynchronous Iterative Methods for Multiprocessors," *J. Assoc. Comp. Mach.* 25, pp. 226-224.

Bokhari, S. [1979]. "On the Mapping Problem," *Proc. Int. Conf. on Par. Proc*, pp. 239-248.

Forsythe, G. and Wasow, W. [1960]. *Finite Difference Methods for Partial Differential Equations*, John Wiley, New York.

Gannon, D. [1981]. "On Mapping non-uniform P.D.E. Structures and Algorithms onto Uniform Array Architectures." *Proc. 1981 Int. Conf. Par. Proc.*, pp. 100-105.

Hayes, L. [1974]. "Comparative Analysis of Iterative Techniques for Solving Laplace's Equation on the Unit Square on a Parallel Processor," M.S. Thesis, Department of Mathematics, University of Texas, Austin.

Jordan, H. [1978]. "A Special Purpose Architecture for Finite Element Analysis," *Proc. 1978 Int. Conf. on Par. Proc.*, pp. 263-266.

Lambiotte, J. [1975]. "The Solution of Linear Systems of Equations on a Vector Computer," Ph.D. Dissertation, University of Virginia.

Ortega, J. and Voigt, R. [1977]. "Solutions of Partial Differential Equations on Vector Computers", *Proc. 1977 Army Num. Anal. Conf.*, pp. 475-526.

Young, D. [1971]. *Iterative Solution of Large Linear Systems*, Academic Press, New York, pp. 427-428.